Operáció rendszerek Bsc

10. Gyakorlat

2022.04.26
készítette:
Czikó Tivadar
Programtervező Informatikus
O2IXLB

Miskolc, 2022

1. Feladat: Adott egy rendszer (foglalási stratégiák), melyben a következő • Szabad területek:
    - 30k, 35k, 15k, 25k, 75k, 45k és
    - Foglalási igények: 39k, 40k, 33k, 20k, 21k állnak rendelkezésre.

A rendszerben a memória 4 kbyte-os blokkokban kerül nyilvántartásra, ennél kisebb méretű töredék igény esetén a teljes blokk lefoglalásra kerül. Határozza meg változó méretű partíció esetén a következő algoritmusok felhasználásával:

first fit, next fit, best fit, worst fit a foglalási igényeknek megfelelő helyfoglalást

– táblázatos formában (az ea. bemutatott mintafeladat alapján)!

Hasonlítsa össze, hogy a teljes szabad memóriaterület hány százaléka vész el átlagosan az egyes algoritmusok esetén! A kapott eredményeket ábrázolja oszlop diagrammal! Magyarázza a kapott eredményeket és hogyan lehet az eredményeket javítani!

First Fit	Szabad területek közül az első szabad méretű						
	Memória terület - Szabad terület						
Foglalási igény	30	35	15	25	75	45	%
39	30	35	15	25	40 | 35	45	33,333333
40	30	35	15	25	35	40 | 5	31,428571
33	30	35	15	25	35	5	100
20	20 | 10	35	15	25	35	5	100
21	10	24 | 11	15	25	35	5	46,666667
Fent maradt	10	11	15	25	35	5	11,111111

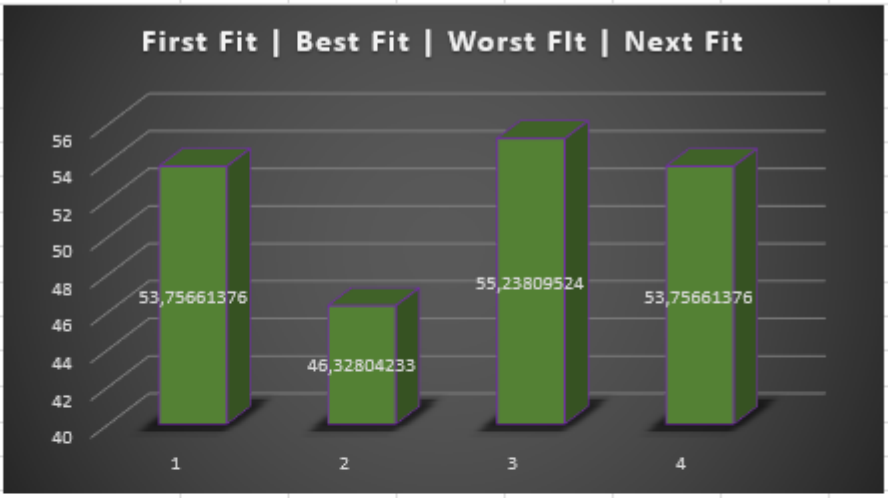
Best Fit	Szabad területek közül az első szabad méretű						
	Memória terület - Szabad terület						
Foglalási igény	30	35	15	25	75	45	%
39	30	20	15	25	40 | 35	45	100
40	30	20	15	25	35	40 | 5	42,857143
33	30	20	15	25	35		100
20	30	35	15	25	40|20|15	5	4
21	30	15	15	24 |1	15	5	20
Fent maradt	30	15	15	1	15	5	11,111111

Worst Fit	Szabad területek közül az első szabad méretű						
	Memória terület - Szabad terület						
Foglalási igény	30	35	15	25	75	45	%
39	30	35	15	25	40 | 35	45	33,333333
40	30	35	15	25	35	45	31,428571
33	30	35	15	25	35	36 | 9	100
20	20 | 10	35	15	25	35	9	100
21	10	24 |11	15	25	35	9	46,666667
Fent maradt	10	11	15	25	35	9	20

Next Fit	Szabad területek közül az első szabad méretű						
	Memória terület - Szabad terület						
Foglalási igény	30	35	15	25	75	45	%
39	30	35	15	25	40 | 35	45	33,333333
40	30	35	15	25	35	40 | 5	31,428571
33	30	35	15	25	35	5	100
20	20 | 10	35	15	25	35		100
21	10	24 | 11	15	25	35		46,666667
Fent maradt:	10	11	15	25	35	5	11,111111

A lilával jelzetet nem lehet kielégiteni!

%-ok átlaga:	53,75661



%-ok átlaga:	46,32804

%-ok átlaga:	55,2381

%-ok átlaga:	53,75661

A foglalási igényt úgy lehetne javítani, ha megemeljük a szabad területek méretét vagy csökkentjük a 33k méretét 32k-ra. Ezzel jelentősen tudnánk csökkenteni az igényét.

2. Feladat: A feladat megoldásához először tanulmányozza Vadász Dénes: Operációs rendszer jegyzet, a témához kapcsolódó fejezetét (6.4)., azaz Írjon C nyelvű programokat, ahol
   • kreál/azonosít szemafor készletet, benne N szemafor-t. A kezdő értéket 0-ra állítja – semset.c,
   • kérdezze le és írja ki a pillanatnyi szemafor értéket – semval.c
   • szüntesse meg a példácskák szemafor készletét – semkill.c
   • sembuf.sem_op=1 értékkel inkrementálja a szemafort – semup.c

```c
/*
 * semset.c
 *
 *  Created on: 2022. ápr. 27.
 *      Author: Tivadar Cziko, O2IXLB
 */

#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>    //Uzenetek, szemaforok es megosztott memoria
#include <sys/sem.h>    //Strukturakat hataroz meg
#include <stdlib.h>

#define KEY 123456L

union semun {
    int val;                  // Value for SETVAL
    struct semid_ds *buf;     // Buffer for IPC_STAT, IPC_SET
    unsigned short *array;    // Array for GETALL, SETALL
    struct seminfo *__buf;    // Buffer for IPC_INFO Linux-specific
};

void main() {
    union semun atr;

    int m = 10;
    int sem = semget(KEY, m, IPC_CREAT | 0666);

    if (sem == -1) {
            perror("Letrehozasa sikertelen!");
        exit(-1);
    }

    atr.array = (short *)calloc(m, sizeof(int));


    if (semctl(sem, 0, SETALL, atr)) {
            perror("Ertek beallitasa sikertelen!\n");
```

```
        exit(-1);
    }
}

/*
 * semset.c
 *
 *  Created on: 2022. ápr. 27.
 *      Author: Tivadar Cziko, O2IXLB
 */

#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>    //Uzenetek, szemaforok es megosztott memoria
#include <sys/sem.h>    //Strukturakat hataroz meg
#include <stdlib.h>

#define KEY 123456L

union semun {
    int val;                // Value for SETVAL
    struct semid_ds *buf;   // Buffer for IPC_STAT, IPC_SET
    unsigned short *array;  // Array for GETALL, SETALL
    struct seminfo *__buf;  // Buffer for IPC_INFO Linux-specific
};

void main() {

    int semID = semget(KEY, 0, 0);
    int n = 10;
    if (semID == -1) {
        perror("Szemaforokat lekerdezese sikertelen!\n");
        exit(-1);
    }

    union semun arg;

    printf("Szemaforok: \n");
    arg.array = (short *)calloc(n, sizeof(int));

    semctl(semID, 0, GETALL, arg);

    for (int i = 0; i < n; i++)
        printf("%d ", arg.array[i]);
}
/*
 * semkill.c
 *
```

```
 *  Created on: 2022. ápr. 27.
 *      Author: Tivadar Cziko, O2IXLB
 */

#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <stdlib.h>
#define KEY 123456L

void main() {
    int n = 5;
    int semID = semget(KEY, 0, 0);
    if (semID == -1) {
        perror("Szemaforokat lekerdezese sikertelen!\n");
        exit(-1);
    }

    for (int i = 0; i < n; i++){
      semctl(semID, i, IPC_RMID);
}
/*
 * semup.c
 *
 *  Created on: 2022. ápr. 27.
 *      Author: Tivadar Cziko, O2IXLB
 */
#include <stdio.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>
#include <stdlib.h>
#define KEY 123456L

void main() {
    int semID = semget(KEY, 0, 0);
    if (semID == -1) {
        perror("Szemaforok lekerdezese sikertelen!\n");
        exit(-1);
    }

    struct sembuf buffer;

    buffer.sem_num = 4;
    buffer.sem_op = 1;
    buffer.sem_flg = 0666;
```

```
    if (semop(semID, &buffer, 1)) {
        perror("Sikertelen\n");
        exit(-1);
    }
```